MPA 634  
Data Science for Managers  
Key Midterm II: Winter 2020

# I. Definitions and Concepts

1. (3 points) First explain how lists are used in R and then explain the function of the “$” notation. Why does mpg$hwy give the highway data from the mpg tibble.

Lists are universal storage objects that are often used in many different computer languages. We use them in R to store heterogeneous information of a variety of shapes and types. Lists contain slots into which we place information. For our class, we have considered logical, integer, double, and character types of data. Lists can also be used to store other lists.

When the slots in a list are named, then we can use the “$” notation to retrieve the values that are in the slot. The mpg$hwy notation finds the slot named “hwy” in the mpg tibble (which is a list) and returns the **values** for highway miles per gallon.

1. (3 points) Use an example to explain the difference between and long and wide tibbles.



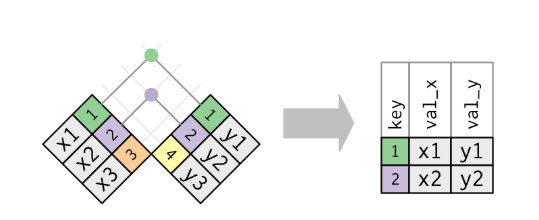
Wide tibbles often include multiple measurements for a given entity. We usually identify the observation with a special identifier such as student in the above example.

Long tibbles use key variables to identify each numerical value. In the above example, the combination of student and exam gives a unique identifier for each score.

1. (3 points) Compare and contrast a left-join with an inner join. Construct an example using two different hypothetical tibbles to illustrate your answer.

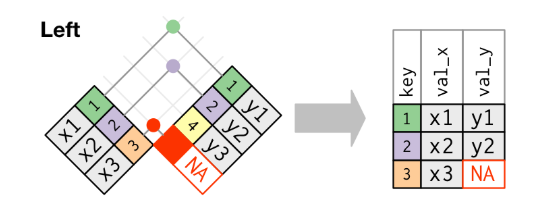
**Inner Join**

An inner join is like an intersection. It includes the rows and values from both of the joining tibbles that share common values for the keys. Any rows where the keys don’t match are discarded. This is illustrated in the diagram from our book that is shown below. In this example, keys match for values one and two but don’t match for three and four. Therefore, only rows one and two are included in the result of the join.



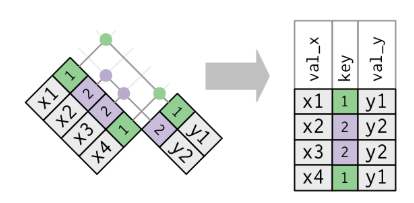
**Left Join**

The left join first completely replicates all the rows in the first tibble. Then it adds information from the second tibble. When the keys between the two tibbles don’t match, then missing values are recorded for the rows which don’t match keys in the second tibble. This is shown in the following diagram:





A powerful result occurs when we have duplicate values for our key. This isn’t required for complete credit on this answer but is information that gives us significant power in practice.



1. (3 points) First define factors and explain what is meant by a factor's level and label.

Factors are categorical variables. The categorical variables can be either nominal or ordinal. The number of categories should be a limited and manageable.

The levels are the possible values for the categories. The levels can be integers or characters. Sometimes, levels are codes that are difficult to remember and interpret. Labels are descriptions that are associated with each level. This makes factors easier to understand and interpret. Statistical software usually prints out the labels rather than levels when reporting the results of an analysis.

1. (3 points) Explain the relationship between the cut function and if else conditional execution.

The cut function is used to create categories based on numerical values in a vector. We first define the ranges for each category. Then we sort each observation into the appropriate category.

The cut function takes a numerical vector and classifies each observation according to the value of the numerical vector. For example, let’s consider the case where we have a vector of ages for our sampled population. We can classify each person as young, middle, or old by the following:



This would be done with

cut(x, breaks = c(0, 29, 64, +Inf), labels = c(“young”, “middle”, “old”), right = TRUE).

Inherent in this logic is conditional execution. If the person is less than 30, they are “young”, if the are 65 or older, then they are “old”. All other ages are “middle”. This logic can be put into an if else framework using the following code:

if(age <= 29){

"young"

} else if (age >= 65){

"old"

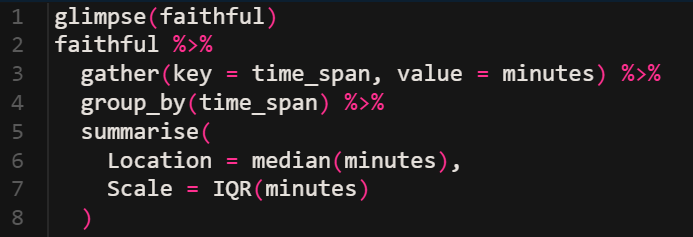
} else{

"middle"

}

# II. Code Interpretation

**Code Chunk I: Interpret lines 3 – 7 (3 points)**

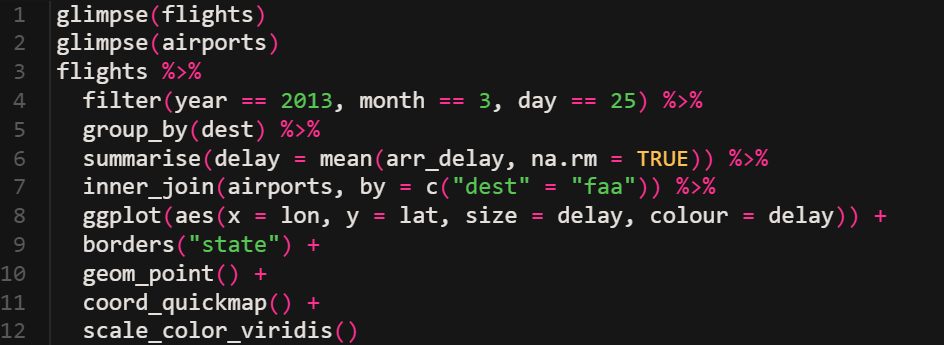


Line 3: Converts a wide tibble into a long tibble. Faithful only contains two variables (eruptions and waiting) so we can stack them on top of each other. The key variable time\_span contains the names of the variables being stacked. The values of the variables are put into a variable called minutes.

Line 4: We anticipate the calculations of the median and interquartile range in the next steps. The group\_by statement allows us to calculate the median and IQR for each of the time\_spans (eruptions and waiting).

Lines 5 – 7: Summarise collapses the tibble into two rows and two columns. We have the location (median) and the scale (interquartile range) for the two variables.

**Code Chunk II: Interpret lines 4 – 7 (4 points)**



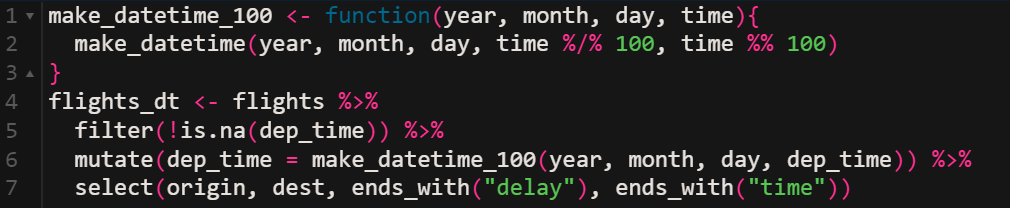
Line 4: Choose those observations for March 25, 2013.

Line 5: We are going to calculate the mean delay for each destination. Therefore, we must group our observations by destination in order for the ensuing mean calculation to work.

Line 6: Creates a new variable called delay, which is the mean of the arrival delay with missing observations removed.

Line 7: Adds the full name of each destination airport. It first replicates the tibble created in lines 3 through 6 and then adds information from the airports tibble where the dest from the new tibble matches the faa variable in the airports tibble. Because this is an inner join, only those rows from both tibbles where the keys match are included.

**Code Chunk III: Interpret lines 1-2 and 5 – 6 (4 points)**

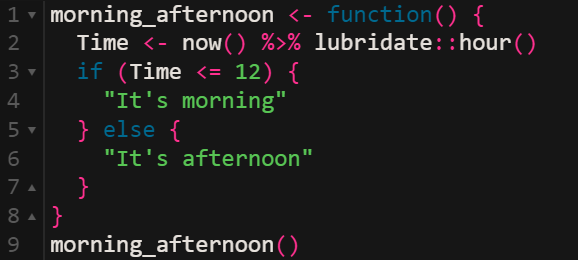


Lines 1 – 2: Creates a new function called make\_datetime\_100. The function has four arguments: year, month, day, and time. Our new function calls the lubridate function make\_datetime. The lubricate function has five arguments: year, month, day, hour, and minute. Since time isn’t really a time variable, we must break in into the hour and minutes that are need for the make\_datetime function. We break the time in the flights tibble into hour and minute by using integer division %/%. This removes the minutes by dividing by 100 and throwing away the remainder. Since the remainder is the number of minutes, the modulus %% operation gives the remainder.

Line 5: Retains only those observations that aren’t missing

Line 6: Replaces the variable named dep\_time with a standard dttm format. This calculation occurs by calling the function that we defined in lines 1 and 2.

**Code Chunk IV: Interpret lines 1 – 6 and 9 (4 points)**



Line 1: Begins the creation of a function called morning\_afternoon which has no arguments.

Line 2: Calculates a new variable called time by using the now() function to first get a date-time from the computer. Next we extract the hour from the date-time by using the hour() function from the lubridate library.

Line 3: Tests the condition of whether the hour stored in the Time variable is less than or equal to 12 o’clock noon.

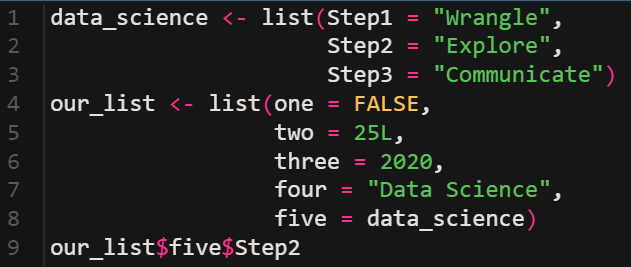
Line 4: If the condition in line 3 is TRUE, then line 4 is executed. Line 4 returns the character string “It’s morning” from the function and exits the function.

Line 5: If the condition in line 3 is FALSE, then the control of the function branches to this line and then executes the next step in line 6.

Line 6: Returns the value “It’s afternoon” from the function and exits the function.

Line 9: Calls the function which executes lines 2 through 8 in the function.

**Code Chunk V: Interpret all lines (4 points)**



Line 1: Initializes a list called data\_science and then names the first slot in the list as Step1 and then injects the value of “Wrangle” into that slot.

Line 2-3: Names the second and third slots in the list “Step2” and “Step3” stores the values “Explore” and “Communicate” respectively.

Line 4: Creates a new list called “our\_list” and then names the first position in the list “one” and stores the logical value “FALSE” in the list.

Lines 5-7: Create named slots “two”, “three”, and “four”. The integer value 25 is assigned to two, the double value “2020” to three, and the character string “Data Science” to four.

Line 8: The slot named “five” contains the list data\_science that was created in lines 1-3.

Line 9: First retrieves the object stored in the slot named five which is the list data\_science. Then the value from the “Step2” slot is retrieved. This returns the value “Explore”.